Google

Web Images Groups News Froogle Local more »

dynamic lexical analyzer Search Advanced Search Preferences

Web Results **1** - **10** of about **37,300** for dynamic lexical analyzer. (**0.55** seconds)

Java Forums - Help on Object (casting)
Sorta, Obj-C does a **dynamic** method disptach everytime you call a method. ...
The **lexical analyzer** is the first phase of a compiler. ...
forum.java.sun.com/thread. jspa?threadID=494628&start=180 - 60k - Cached - Similar pages

Comp Review Spring 1996: Compilers
Why separate **lexical analysis** phase? 1. Simpler lang design parser processing
comments and ... Static Vs **Dynamic** type checking Structural equivalence (p. ...
www.personal.kent.edu/ ~rmuhamma/Compilers/compreview.html - 17k - Cached - Similar pages

CodeGuru: A fast **lexical analyzer** with IDE
A library containing an **lexical analyzer**, including an IDE for developing rule sets.
... Need a **dynamic** range of support solutions for Linux? WHITEPAPER : ...
www.codeguru.com/Cpp/misc/misc/article.php/c3835/ - 60k - Aug 30, 2005 - Cached - Similar pages

The **Lexical Analyzer**
Next: The **Dynamic** Structure Up: Syntax and Data Structure Previous: Implementation
... The **lexical analyzer** is not a DFA. At the development phase the lex ...
www.ceng.metu.edu.tr/~ucoluk/ research/lisp/lispman/node26.html - 4k - Cached - Similar pages

[PDF] FLEX(1) FLEX(1) flex − fast **lexical analyzer** generator
File Format: PDF/Adobe Acrobat - View as HTML
and delete for creating and destroying **dynamic** objects. Finally, the YY_CURRENT_BUFFER
macro ... ME Lesk and E. Schmidt, LEX − **Lexical Analyzer** Generator ...
homepages.inf.ed.ac.uk/stark/ipp/manuals/flex.pdf - Similar pages

Bookpool: Compilers: Principles, Techniques, and Tools
The role of the **lexical analyzer**; Input buffering; Specification of tokens ...
Dynamic storage allocation techniques; Storage allocation in Fortran ...
www.bookpool.com/sm/0201100886 - 13k - Cached - Similar pages

CMPSC 160 Course Description
Lexical analyzer generators, JLex **lexical analyzer** generator. ... Type checking,
type systems, type expressions, static vs. **dynamic** type checking, ...
www.cs.ucsb.edu/~bultan/courses/160/ - 10k - Cached - Similar pages

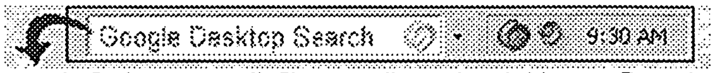
Lexical analysis in source code scanning - Uninet Infosec 02
Dynamic analysis, in constrast, takes the whole or parts of the program and runs
... A **lexical analyzer** looks at the patterns as they flow by and performs ...
www.monkey.org/~jose/presentations/ czech-rubicon02.d/czech.html - 47k - Cached - Similar pages

C++ File Checklist
... efsm.l: Input file for lex utility to create EFSM **lexical analyzer**; efsmll.cpp:
EFSM **lexical analyzer** code generated by lex utility ...
www.site.uottawa.ca/~tunguyen/TSR/Cfiles.html - 6k - Cached - Similar pages

Syllabus-98
Chapter 3 - **Lexical Analysis**. 3.1 The role of the **lexical analyzer**. 3.2 Input

buffering ... 7.7 Language facilities for **dynamic** storage allocation ...
www.cs.brandeis.edu/~cs31b/Syllabus-98.htm - 6k - Cached - Similar pages

Goooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**



Free! Instantly find your email, files, media and web history. Download now.

dynamic lexical analyzer | Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Web Images Groups News Froogle Local **more »**

run-time lexical analyzer Search Advanced Search Preferences

Web Results **1** - **10** of about **33,500** for **run-time lexical analyzer**. (**0.21** seconds)

Did you mean: ***runtime*** lexical analyzer

2. **Lexical analysis**
This chapter describes how the **lexical analyzer** breaks a file into tokens. ...
The **run-time** character set depends on the I/O devices connected to the ...
docs.python.org/ref/**lexical**.html - 8k - Cached - Similar pages

The Code Project - CLex: A Programmable **Lexical Analyser** - C++ / MFC
Create a **lexical analyzer** on the fly by constructing with a lex specification.
... This is achieved by the construction of a state machine at **runtime** from ...
www.codeproject.com/cpp/BHLex.asp - 49k - Cached - Similar pages

CMPSC 160 Course Description
... **lexical analysis**; syntax **analysis** including LL and LR parsers; type checking;
run-time ... **Lexical analyzer** generators, JLex **lexical analyzer** generator. ...
www.cs.ucsb.edu/~bultan/courses/160/ - 10k - Cached - Similar pages

[PDF] FLEX(1) FLEX(1) flex − fast **lexical analyzer** generator
File Format: PDF/Adobe Acrobat - View as HTML
... if a call to unput() results in too much text being pushed back; instead, a
run-time error ... ME Lesk and E. Schmidt, LEX − **Lexical Analyzer** Generator ...
homepages.inf.ed.ac.uk/stark/ipp/manuals/flex.pdf - Similar pages

Arcadia: CU Arcadia: Java Bison Parser **Runtime**
It is the primary **runtime** parse engine. It contains one class: yyparse. ...
It contains a subclass of yylex to do the actual **lexical analysis** appropriate to ...
serl.cs.colorado.edu/~arcadia/Software/jb.html - 9k - Cached - Similar pages

Lex - A **Lexical Analyzer** Generator
The **lexical analysis** programs written with Lex accept ambiguous ... Compatible
run-time libraries for the different host languages are also provided. ...
www.combo.org/lex_yacc_page/lex.html - 56k - Cached - Similar pages

JAVA-LEX & CUP Tutorial
The name of the **lexical analyzer** file will be the name of the Java-Lex ...
By definition, the parser tokens must be of java_cup.**runtime**.token type or a ...
www.hio.hen.nl/~vanleeuw/pse/spanje/tutorial.html - 16k - Cached - Similar pages

lex(1): fast **lexical analyzer** generator - Linux man page
... to unput() results in too much text being pushed back; instead, a **run-time**
error results. ... ME Lesk and E. Schmidt, LEX - **Lexical Analyzer** Generator ...
www.die.net/doc/linux/man/man1/lex.1.html - 112k - Cached - Similar pages

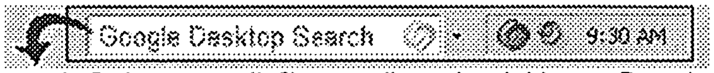
Three Java Variants Extend the Language
... at **run-time** which mixins to merge into your base class. **Lexical Analysis** ...
You can do this by overriding one of the methods of the **lexical analyzer**, ...
www.devx.com/java/Article/10471/1763/page/4 - 20k - Cached - Similar pages

JLex:A **lexical analyzer** generator for Java
The JLex utility is based upon the Lex **lexical analyzer** generator model. ...
%type java_cup.**runtime**.Symbol See the next section for more details on these ...
www.iam.unibe.ch/~fki/lectures/CO/JLexManual.html - 48k - Cached - Similar pages

Did you mean to search for: ***runtime*** lexical analyzer

Goooooooooogle ▶

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**



Free! Instantly find your email, files, media and web history. Download now.

run-time lexical analyzer Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

lexical analyzer dynamic

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **lexical analyzer dynamic** Found **6,819** of **160,906**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Systematically derived instruction sets for high-level language support
Pradip Bose, B. R. Rau, M. S. Schlansker
April 1982 **Proceedings of the 20th annual Southeast regional conference**
Full text available: pdf(729.75 KB) Additional Information: full citation, abstract, references, citings

Conventional machine-languages (instruction sets) were not designed with high-level languages (HLLs) in mind. The resulting semantic gap is known to cause significant inefficiencies in program representation and execution time. Direct interpretation of HLLs is not the solution, because it is too complex and inefficient. The alternative is to precede the interpretation phase by a compilation phase in which the HLL is translated to a "suitable" intermediate representation which is directly interpr ...

Keywords: compilation, directly interpretable languages, high-level languages, instruction set design, interpretation, semantic gap, space-time efficiency, syntax and semantics

2 Lightweight lexical source model extraction
Gail C. Murphy, David Notkin
July 1996 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 5 Issue 3
Full text available: pdf(364.49 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Software engineers maintaining an existing software system often depend on the mechanized extraction of information from system artifacts. Some useful kinds of information—source models—are well known: call graphs, file dependences, etc. Predicting every kind of source model that a software engineer may need is impossible. We have developed a lightweight approach for generating flexible and tolerant source model extractors from lexical specifications. The approach is lightweight ...

Keywords: lexical analysis, lexing, reverse engineering, scanner generation, scanning, software maintenance, source code analysis, source model, static analysis

3 Migration of legacy web applications to enterprise Java™ environments net.data® to JSP™ transformation
Yu Ping, Jianguo Lu, Terence C. Lau, Kostas Kontogiannis, Tack Tong, Bo Yi
October 2003 **Proceedings of the 2003 conference of the Centre for Advanced Studies**

on Collaborative research
Full text available: pdf(165.69 KB) Additional Information: full citation, abstract, references, index terms

As Web technologies advance, the porting and adaptation of existing Web applications to take advantage of the advancement has become an issue of increasing importance. Examples of such technology advancement include extensible architectural designs, more efficient caching protocols, and provision for customizable dynamic content delivery. This paper presents an experience report on the migration of legacy IBM® Net.Data® based applications to new enterprise Java
Keywords: Java 2 Enterprise Edition (J2EE™), JavaBeans, JavaServer pages, Net.Data, SQL, migration, model-view-controller (MVC), transformation

4 Experience with an experimental compiler generator based on denotational semantics
James Bodwin, Laurette Bradley, Kohji Kanda, Diane Litle, Uwe Pleban
June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction**, Volume 17 Issue 6
Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings, index terms

Compiler generation based on formal semantics has received considerable attention in recent years from a number of semanticists. Compiler writers, on the other hand, know relatively little about these efforts. This paper tries to remedy this situation by discussing our experimentation with the Semantics Implementation System (SIS) of Peter Mosses. SIS allows the user to generate a complete compiler from a formal specification of the syntax and semantics of a programming language. In particu ...

5 Design of instruction set architectures for support of high-level languages
Pradip Bose, Edward S. Davidson
January 1984 **ACM SIGARCH Computer Architecture News , Proceedings of the 11th annual international symposium on Computer architecture**, Volume 12 Issue 3
Full text available: pdf(795.07 KB) Additional Information: full citation, abstract, references, citings, index terms

Conventional instruction sets or directly interpretable languages (DILs) have not been designed with high-level languages (HLLs) in mind. The modern design problem is to derive a space-time efficient DIL for a HLL processing system. In this paper, we present our approach to the problem of designing well-matched, space-time efficient DILs. A systematic, syntax- and semantics-directed DIL design methodology is presented. It calls for an incremental transformation of the source HLL, until a su ...

6 Code generation using tree matching and dynamic programming
Alfred V. Aho, Mahadevan Ganapathi, Steven W. K. Tjiang
October 1989 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 11 Issue 4
Full text available: pdf(1.87 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Compiler-component generators, such as lexical analyzer generators and parser generators, have long been used to facilitate the construction of compilers. A tree-manipulation language called twig has been developed to help construct efficient code generators. Twig transforms a tree-translation scheme into a code generator that combines a fast top-down tree-pattern matching algorithm with dynamic programming. Twig has been used to specify an ...

7
A Partitioning Methodology for Accelerating Applications in Hybrid Reconfigurable Platforms

M. D. Galanis, A. Milidonis, G. Theodoridis, D. Soudris, C. E. Goutis
March 2005 **Proceedings of the conference on Design, Automation and Test in Europe - Volume 3**
Full text available: pdf(138.88 KB) Additional Information: full citation, abstract

In this paper, we propose a methodology for partitioning and mapping computational intensive applications in reconfigurable hardware blocks of different granularity. A generic hybrid reconfigurable architecture is considered so as the methodology can be applicable to a large number of heterogeneous reconfigurable platforms. The methodology mainly consists of two stages, the analysis and the mapping of the application onto fine and coarse-grain hardware resources. A prototype framework consisting ...

8 THALES: a software package for plane geometry constructions with a natural language interface
K. Fábricz, Z. Alexin, T. Gyimóthy, T. Horváth
August 1990 **Proceedings of the 13th conference on Computational linguistics - Volume 1**
Full text available: pdf(301.09 KB) Additional Information: full citation, abstract, references

THALES is a software package for plane geometry constructions, supplied with a natural language interface. Using THALES requires no knowledge of a programming language. The interface is capable of processing practically all kinds of instructions within the subset of plane geometry English. The "static semantic" module has been generated on the basis of a high-level attribute specification. Transportability, modifiability and generality -- the key issues of natural language interface design -- ar ...

9 Bootstrapping morphological analyzers by combining human elicitation and machine learning
Kemal Oflazer, Sergei Nirenburg, Marjorie McShane
March 2001 **Computational Linguistics**, Volume 27 Issue 1
Full text available: pdf(1.76 MB) Publisher Site Additional Information: full citation, abstract, references

This paper presents a semiautomatic technique for developing broad-coverage finite-state mor-phological analyzers for use in natural language processing applications. It consists of three components---elicitation of linguistic information from humans, a machine learning bootstrapping scheme, and a testing environment. The three components are applied iteratively until a threshold of output quality is attained. The initial application of this technique is for the morphology of low-density languag ...

10 Using Ada 95 in a compiler course
S. Tucker Taft
September 2001 **ACM SIGAda Ada Letters , Proceedings of the 2001 annual ACM SIGAda international conference on Ada**, Volume XXI Issue 4
Full text available: pdf(22.94 KB) Additional Information: full citation, abstract, index terms

In this extended abstract, we describe the use of Ada 95 in a Compiler Construction Course.

Keywords: Ada 95, compiler construction, course

11 Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science
William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J.

Schweppe, William Viavant, David M. Young
March 1968 **Communications of the ACM**, Volume 11 Issue 3
Full text available: pdf(6.63 MB) Additional Information: full citation, references, citings

Keywords: computer science academic programs, computer science bibliographies, computer science courses, computer science curriculum, computer science education, computer science graduate programs, computer science undergraduate programs

12 Ontological semantics, formal ontology, and ambiguity
Sergei Nirenburg, Victor Raskin
October 2001 **Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001**
Full text available: pdf(1.22 MB) Additional Information: full citation, abstract, references, citings, index terms

Ontological semantics is a theory of meaning in natural language and an approach to natural language processing (NLP) which uses an ontology as the central resource for extracting and representing meaning of natural language texts, reasoning about knowledge derived from texts as well as generating natural language texts based on representations of their meaning. Ontological semantics directly supports such applications as machine translation of natural languages, information extraction, ...

13 Systems: University of Manitoba: description of the NUBA system as used for MUC-5
Dekang Lin
August 1993 **Proceedings of the 5th conference on Message understanding MUC5 '93**
Full text available: pdf(762.71 KB) Additional Information: full citation, abstract, references

Abduction is the inference to the best explanation. Many tasks in natural language understanding such as word-sense disambiguity [1], local pragmatics [4], metaphor interpretation [3], and plan recognition [5, 8], can be viewed as abduction.

14 Toolpack - an experimental software development environment research project
Leon J. Osterweil
September 1982 **Proceedings of the 6th international conference on Software engineering**
Full text available: pdf(869.04 KB) Additional Information: full citation, abstract, references, index terms

This paper describes a research project aimed at building and studying prototype environments for Mathematical Software. The project is aimed at gaining actual measurements and experiences that should help solidify knowledge about how to build effective environments. Towards this end some speculative ideas about environment file systems and command languages are presented, along with research plans for effectively evaluating these and other design notions.

15 AUTO STAR—a software development system
M. Y. Zhu
March 1989 **ACM SIGPLAN Notices**, Volume 24 Issue 3
Full text available: pdf(883.50 KB) Additional Information: full citation, abstract, citings, index terms

This report summarizes the AUTO STAR project. AUTO STAR is a software development system which accepts the algebraic specification of a given software system and produces an Ada implementation of that system."In many applications, algorithms play almost no role, and certainly present almost no problem. The real problem is the mass of detailed requirements; and the only solution is the discovery or invention of general rules and abstractions which cover the many thousands of cases with as few exc ...

16 Morphology & tagging: An efficient treatment of Japanese verb inflection for morphological analysis
Toru Hisamitsu, Yoshihiko Nitta
August 1994 **Proceedings of the 15th conference on Computational linguistics - Volume 1**
Full text available: pdf(557.85 KB) Additional Information: full citation, abstract, references

Because of its simple appearance, Japanese verb inflection has never been treated seriously. In this paper we reconsider traditional lexical treatments of Japanese verb inflection, and propose a new treatment of verb inflection which uses newly devised segmenting units. We show that our proposed treatment minimizes the number of lexical entries and avoids useless segmentation. It requires 20 to 40% less chart parsing computation and it is also suitable for error correction in optical character r ...

17 Lexical analysis using table look-up (abstract)
Brian Smith, Dominic Soda, George W. Zobrist
February 1986 **Proceedings of the 1986 ACM fourteenth annual conference on Computer science**
Full text available: pdf(66.73 KB) Additional Information: full citation, references

18 Space-efficient closure representations
Zhong Shao, Andrew W. Appel
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming**, Volume VII Issue 3
Full text available: pdf(1.26 MB) Additional Information: full citation, abstract, references, citings, index terms

Many modern compilers implement function calls (or returns) in two steps: first, a closure environment is properly installed to provide access for free variables in the target program fragment; second, the control is transferred to the target by a "jump with arguments (or results)". Closure conversion, which decides where and how to represent closures at runtime, is a crucial step in compilation of functional languages. We have a new algorithm t ...

19 Memory subsystem performance of programs using copying garbage collection
Amer Diwan, David Tarditi, Eliot Moss
February 1994 **Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Heap allocation with copying garbage collection is believed to have poor memory subsystem performance. We conducted a study of the memory subsystem performance of heap allocation for memory subsystems found on many machines. We found that many machines support heap allocation poorly. However, with the appropriate memory subsystem organization, heap allocation can have good memory subsystem performance.

20 Applications: Parsing and case analysis in TANKA
Terry Copeck, Sylvain Delisle, Stan Szpakowicz
August 1992 **Proceedings of the 14th conference on Computational linguistics - Volume 3**
Full text available: pdf(422.73 KB) Additional Information: full citation, abstract, references

The TANKA project seeks to build a model of a technical domain by semi-automatically processing unedited English text that describes this domain. Each sentence is parsed and conceptual elements are extracted from the parse. Concepts are derived from the Case structure of a sentence, and added to a conceptual network that represents knowledge about the domain. The DIPETT parser has a particularly broad coverage of English syntax; its newest version can also process sentence fragments. The HAIKU s ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next